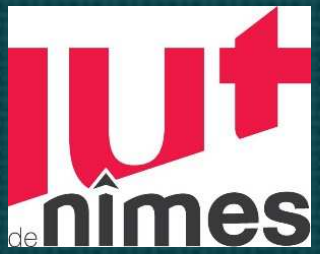


Notions d'algorithmique et de programmation

Environnement

Visual Basic sous Excel XP

Christophe PEYRE

Travail à automatiser

ANALYSE

Description précise et rigoureuse du traitement

ALGORITHME

Langage de programmation

langage machine (binaire)

Programme : Suite d'**instructions** exécutables par l'ordinateur.

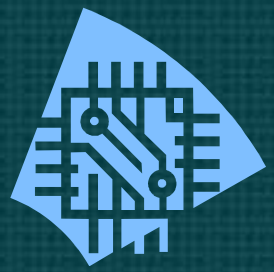
Différents niveaux de complexité des programmes

- Affichage, tri, saisie d'informations:
  « Routines de traitement »
- Fonctions de calcul
- Librairies
- Programmes de traitement (saisie, calculs, éditions des résultats)
- Logiciels (Ensembles de programmes associés)

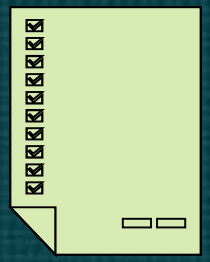
Pourquoi cet enseignement?

- Automatisation de tâches répétitives
- Enrichissement de la bibliothèque de fonctions d ’Excel
- Apport pédagogique:
  - Structuration et organisation d’un raisonnement
  - Précision, rigueur
- Capacité de dialogue avec les professionnels
- Prise de conscience des limites et des possibilités de l’outil

Langages de programmation



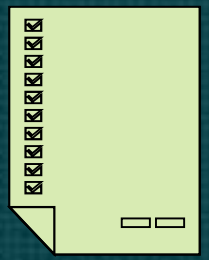
Au cœur du processeur : langage « machine », lié au processeur



Programme en Assembleur

(lié au processeur)

Traduction par un programme intermédiaire



Programme en langage de haut niveau, interprété ou compilé

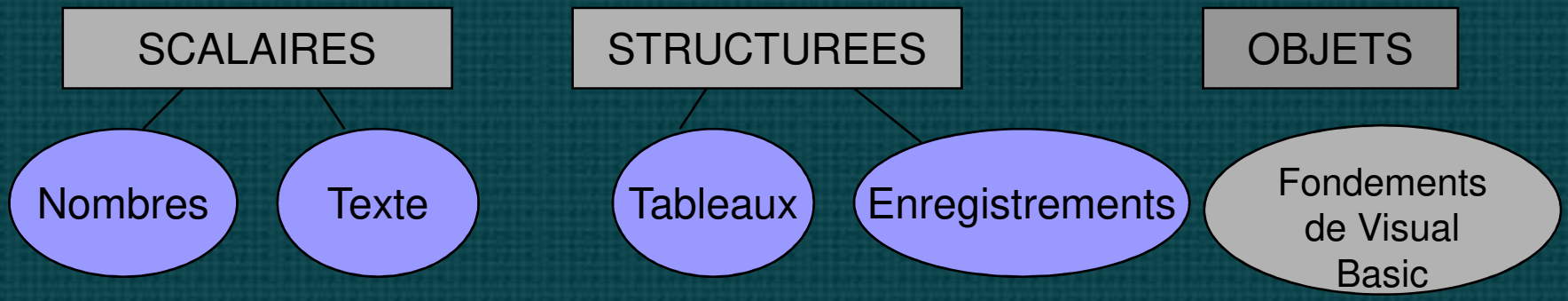
Pascal, C++, C#, Fortran, Basic, Lisp, Java,..

Langages de programmation (2)

- Assembleur: Gestion périphériques de l'ordinateur, calculs fortement optimisés,..
- Langages de haut niveau : Pas de gestion particulière du matériel: Indépendance vis-à-vis de la « plateforme » (Mac-OS, Linux, Windows).
- Dans tous les cas, la syntaxe doit être RIGOUREUSEMENT RESPECTEE.
- De nombreux POINTS COMMUNS existent entre les différents langages.

Eléments principaux du langage

- Données à traiter



- Moyens de traitement: **instructions** composant le **code** exécutable

Données à traiter

- Manipulées individuellement ou en groupe

Structurées

Un identificateur désigne un ensemble de données :

- de même nature (tableaux)
- de nature différente (enregistrements)

Scalaires

Un identificateur désigne une seule donnée :

Nombre (entier, décimal)
Valeur logique (VRAI, FAUX)
Caractère
Objet
Chaîne de caractères

Variables et Constantes

- Les données ont une durée de vie :
  - Une constante est en général définie au début d'un programme, son contenu est fixe. Ex : Const. Pi as single
  - Le contenu d'une variable peut être « écrasé » par un nouveau

$A \leftarrow A + 1$

+1= 3

Caractéristiques d'une variable

Variable = Emplacement en mémoire vive

Doit être **DÉCLARÉE :**

- **Identificateur** (son nom)
- **TYPE** (scalaire ou structurée)

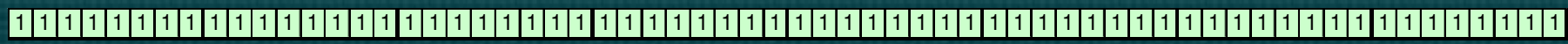
Repérage de l'emplacement

Taille (nombre d'octets) de celui-ci

Une variable a une portée globale (accessible par tous les modules du prg) ou locale

- Portée « Procédure unique » : Déclarer une instruction « **Dim** », Static ou Private devant la variable : Déclaration dans la procédure
- Portée « Module unique » : Déclarer une instruction « **Dim** » devant la variable avant la première procédure : Déclaration dans le module
- Portée « Tous les Modules » : Déclarer un instruction « **Public** » devant la variable avant la première procédure : Déclaration dans n'importe quel module

Types de données scalaires « classiques »



- Entier : **Integer** (Adressage mémoire : **2 octets)**
- Réel simple précision : **Single** (Adressage mémoire : **4 octets)**
- Réel double précision : **Double** (Adressage mémoire : **8 octets)**
- Octet : **Byte** (Adressage mémoire : **1 octets)**
- Chaîne de caractères : **String** (Adressage mémoire : **x octets)**

Types de données scalaires « classiques »

Nom	Type	Détails	Symbole
Byte (Octet)	*Numérique*	Nombre entier de 0 à 255.	
Integer	*Numérique*	Nombre entier de -32'768 à 32'767.	%
Long	*Numérique*	Nombre entier de - 2'147'483'648 à 2'147'483'647.	&
Currency	*Numérique*	Nombre à décimale fixe de -922'337'203'685'477.5808 à 922'337'203'685'477.5807.	@
Single	*Numérique*	Nombre à virgule flottante de -3.402823E38 à 3.402823E38.	!
Double	*Numérique*	Nombre à virgule flottante de -1.79769313486232D308 à 1.79769313486232D308.	#
String	*Texte*	Texte.	$
Date	*Date*	Date et heure.	
Boolean	*Boolean*	True (vrai) ou False (faux).	
Object	*Objet*	Objet Microsoft.	
Variant	*Tous*	Tout type de données (type par défaut si la variable n'est pas déclarée).	

Types de données scalaires « classiques »

```
'Exemple : nombre entier
Dim nbEntier As Integer
nbEntier = 12345

'Exemple : nombre à virgule
Dim nbVirgule As Single
nbVirgule = 123.45

'Exemple : texte
Dim varTexte As String
varTexte = "Excel-Pratique.com"

'Exemple : date
Dim varDate As Date
varDate = "06.02.2011"

'Exemple : vrai/faux
Dim varBoolean As Boolean
varBoolean = True

'Exemple : objet (objet Worksheet pour cet exemple)
Dim varFeuille As Worksheet
Set varFeuille = Sheets("Feuil2") 'Set => attribution d'une valeur à une variable objet

    'Exemple d'utilisation de la variable objet : activation de la feuille
    varFeuille.Activate
```

Données à traiter

- Exemples de données structurées

Enregistrement B

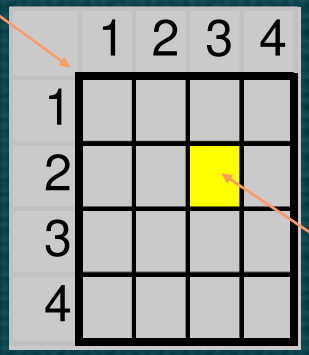
B

CODE	*Z-235*
Qté	*3*
Unité	*M2*
Prix Unitaire	*345,67*

B.Unité

A

Tableau A



Désignation d'un élément:

A (2,3)

Cas des objets Excel

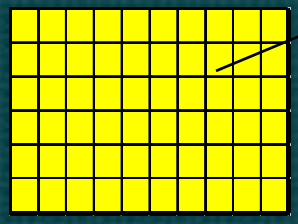
- Excel : C'est l'objet Application qui contient d'autres objets : Ces objets sont :
- Caractérisés par :
  - Des propriétés
  - Des méthodes : Actions que l'on peut effectuer avec les objets
  - Des gestionnaires d'événements
- Exemples d'objets (VBA – Excel) :
  - Classeur : « Workbook »
  - Feuille de calcul « Worksheet »
  - Plage de cellule « Range »

Exemple d'objet sous Excel : Objet RANGE

Les objets possèdent des :

Propriétés

- Address
- Value
- Borders
- ColumnWidth
- Font
- NumberFormat
- Etc…

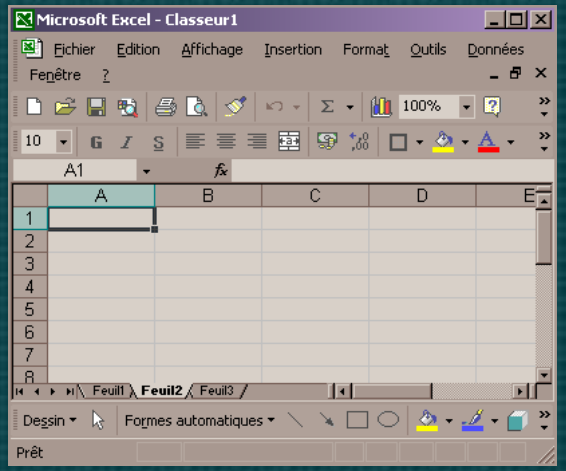


Plage de cellules:
Type RANGE

Méthodes

- BorderAround
- ClearContents
- Copy
- Delete
- Sort
- Etc…

Exemple d'objet sous Excel : Objet Workbook (Classeur)



Propriétés

- FullName
- HasPassword
- Saved
- Sheets
- Worksheets
- Etc…

Méthodes

- Close
- Protect
- SaveAs
- Delete
- Etc…

Événements

- BeforeClose
- BeforePrint
- Open
- SheetChange
- Etc…

Collections d'objets

« Groupes d'objets de la même Classe »

- Les objets peuvent être regroupés en collections :
  - Worksheets : *Ensemble des feuilles de calcul* Worksheet *d'un classeur*
  - Sheets : *Ensemble des feuilles calculs et graphiques*
  - Workbooks : *Ensemble des classeurs* Workbook *ouverts en mémoire*
  - Charts: *Ensemble des feuilles graphiques du classeur*
- Exemples de désignation de l'un des éléments :
  - Worksheets("Feuil1") : *feuille N°1 de la collection des feuilles du classeur*
  - Worksheets(1) *si feuille 1 est la 1ère dans la collection*
  - Charts(1).Type : *Type de graphe de celui qui est situé dans la première feuille graphique*

Objets (suite)

- Référencer des objets : Pour référencer un objet :
- Qualifier l'objet en reliant les noms d'objet par des points

Workbooks(''Classeur1'').Worksheets(''Feuil1'')

Par ex : si 2 classeurs sont ouverts sinon :

Worksheets(''Feuil1'') : VBA recherche Feuil1 dans le classeur actif

Workbooks(''Classeur1'').Worksheets(''Feuil1'').Range(''A1'') : *Pour référencer une cellule de la ''Feuil1'' du ''Classeur1''*

- Référence complète pour l'exemple précédent :

Application.Workbooks(''Classeur1'').Worksheets(''Feuil1'').Range(''A1'')

→ *Qui devient si ''Classeur1'' est actif :*

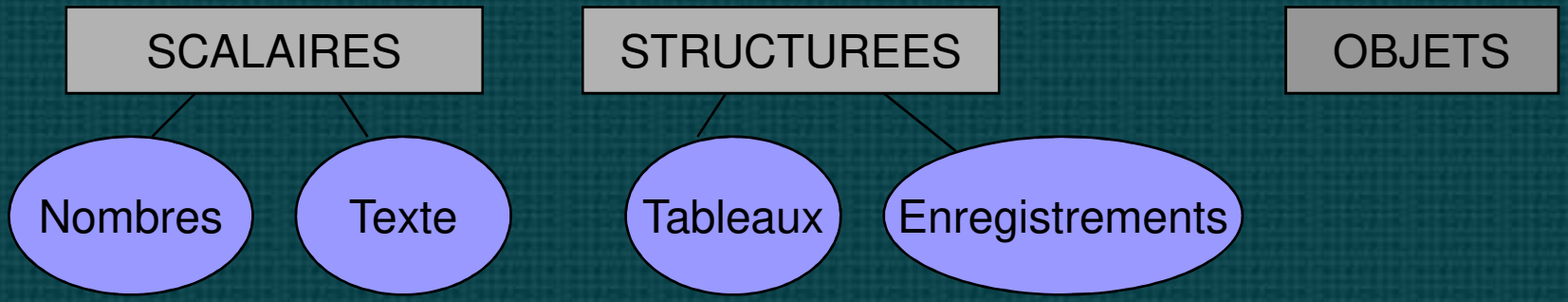
Worksheets(''Feuil1'').Range(''A1'')

→ *Qui devient si ''Feuil1'' est active :*

Range(''A1'')

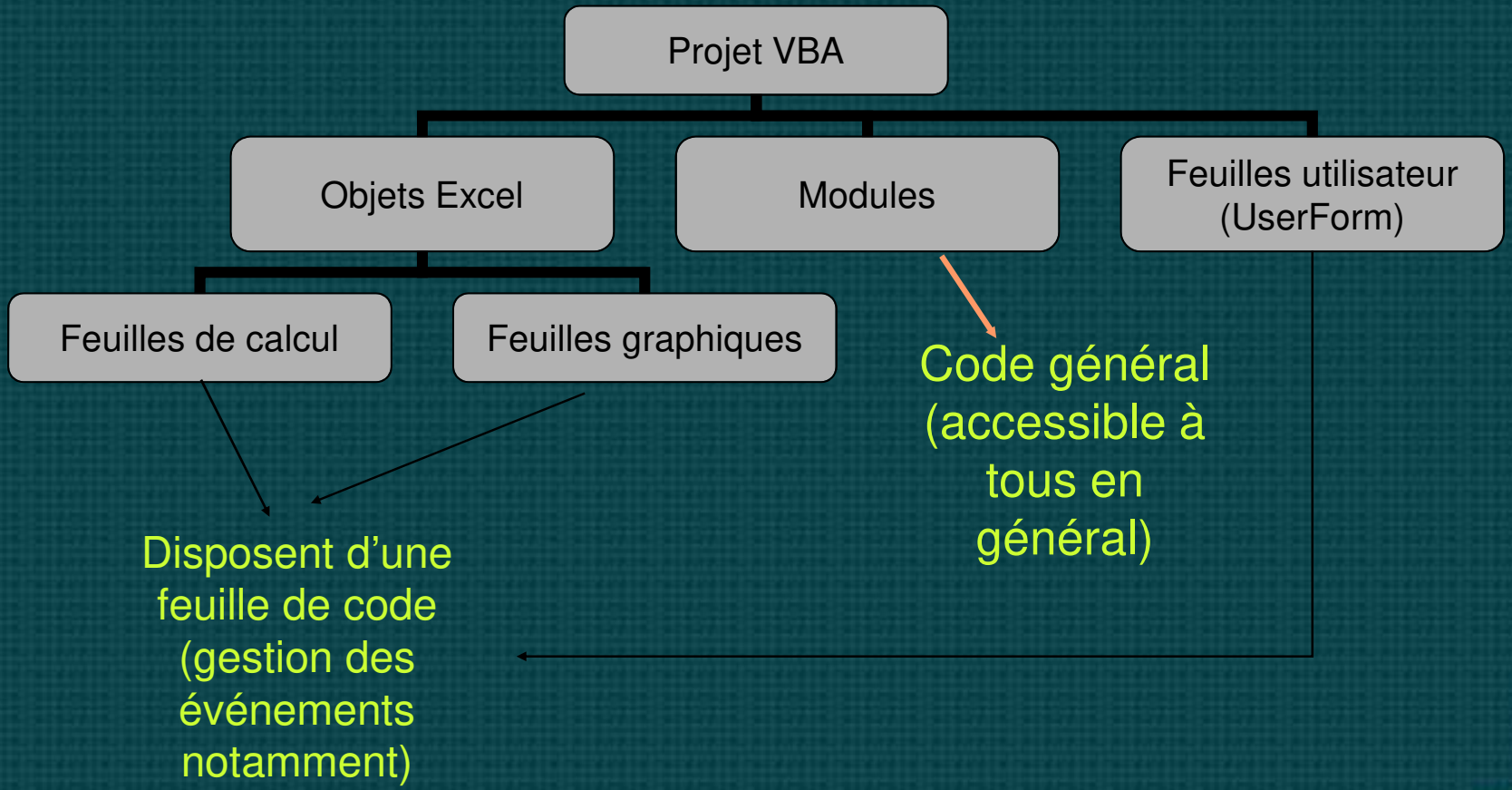
Eléments principaux du langage

- Données à traiter

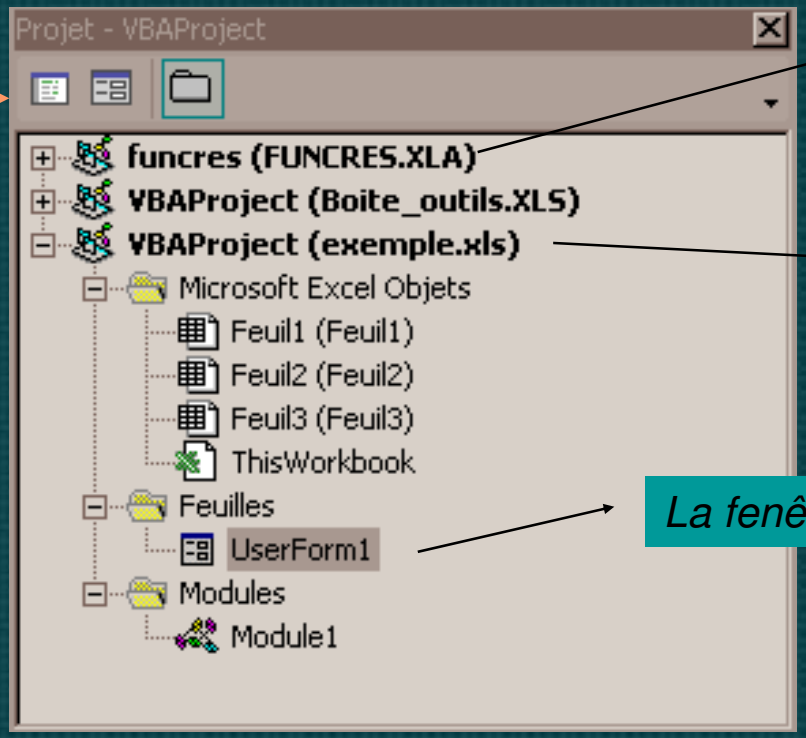


- Moyens de traitement : ***Instructions*** composant le **code** exécutable

Moyens de traitement : Organisation générale d'un Projet « VBAProject »



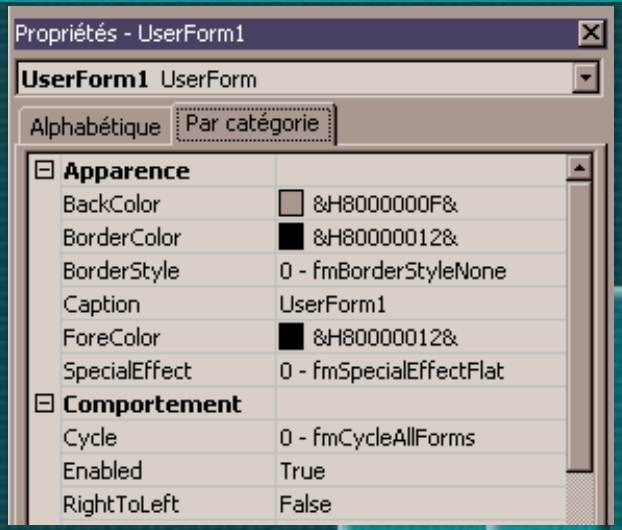
Environnement VBA : Gestionnaire de projet



Macro complémentaire utilitaire analyse

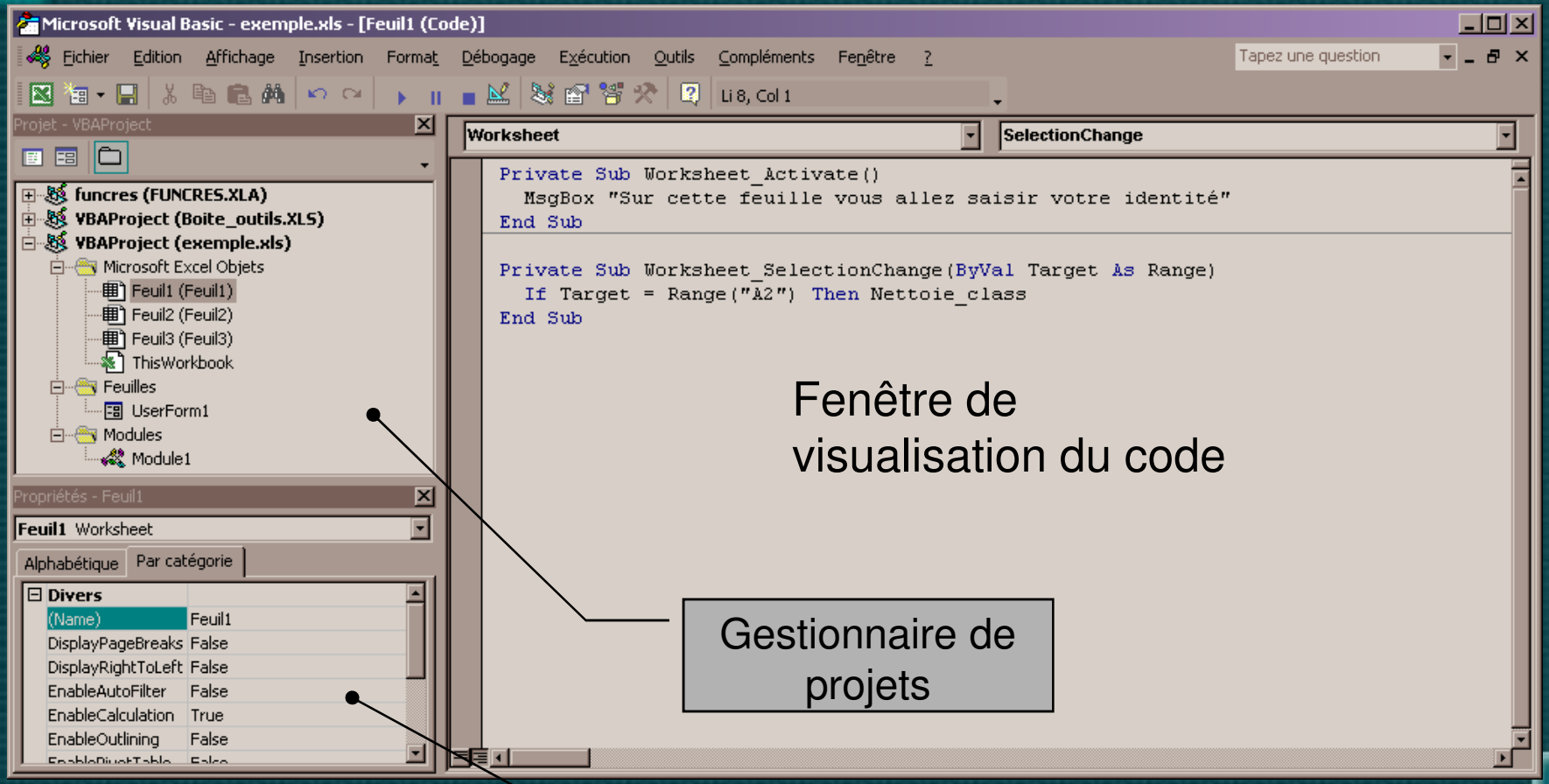
Classeurs ouverts : Projets

La fenêtre Propriétés concerne l'élément sélectionné



Ici on décide d'afficher l'objet ou bien le code associé

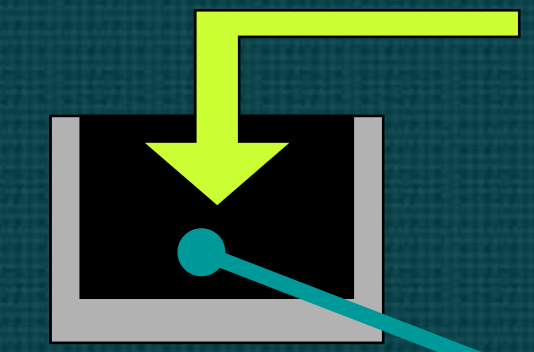
Environnement VBA: Vue d'ensemble



Moyens de traitement

- Code composé d'**instructions**
- Instructions regroupées sous la forme de **procédures**
  - **Procédure :** séquence d'instructions s'exécutant en tant qu'entité et résidant dans un module VBA
  - Un seul module peut contenir un nombre quelconque de procédures
- **Trois instructions de base dans les procédures:**
  - **DECLARATIONS** (Voir les exemples directement en VBA)
  - **AFFECTATIONS**
  - **APPEL (exécution) de PROCÉDURE (Sub) ou (Function) ou (Property)** (Voir les exemples directement en VBA)

Instruction de base : Affectation



Expression (calcul) fournissant un résultat d'un type compatible avec celui de A

A ← Expression

A

Variable

Propriété d'un objet

Ancien contenu : écrasé par le nouveau

Instruction de base: Affectation (2)

A ← Expression

Pour tous les langages :

- La valeur obtenue dans le terme de droite est stockée dans la variable (ou la propriété) désignée à gauche
- L'expression fait intervenir des opérateurs (+,-,/,*), des opérandes (constantes, identificateurs, fonctions)

(2) : Traduction du symbole général d'affectation (flèche) :

Basic, C, Java, PHP, Javascript : **=**

Pascal: **:=**

Opérateurs (utilisés en VBA), expressions

Opérateurs de calcul	
Addition	+
Soustraction	-
Multiplication	*
Division	/

Expressions de type numérique

Fonctions mathématiques

Concaténation de chaines	
Basic, Pascal, C	**PHP**
+	**.**

Expressions de type chaîne

Fonctions de traitement de chaînes

Opérateurs de comparaison		
	Basic, Pascal	**Javascript, Java, C++**
égalité	**=**	**==**
inférieur	**<**	**<**
supérieur	**>**	**>**
différent	**<>**	**!=**

Opérateurs logiques		
	Basic, Pascal	**Javascript, Java, C++**
OU	**OR**	**||**
ET	**AND**	**&&**
NON	**NOT**	**!**

Expressions logiques

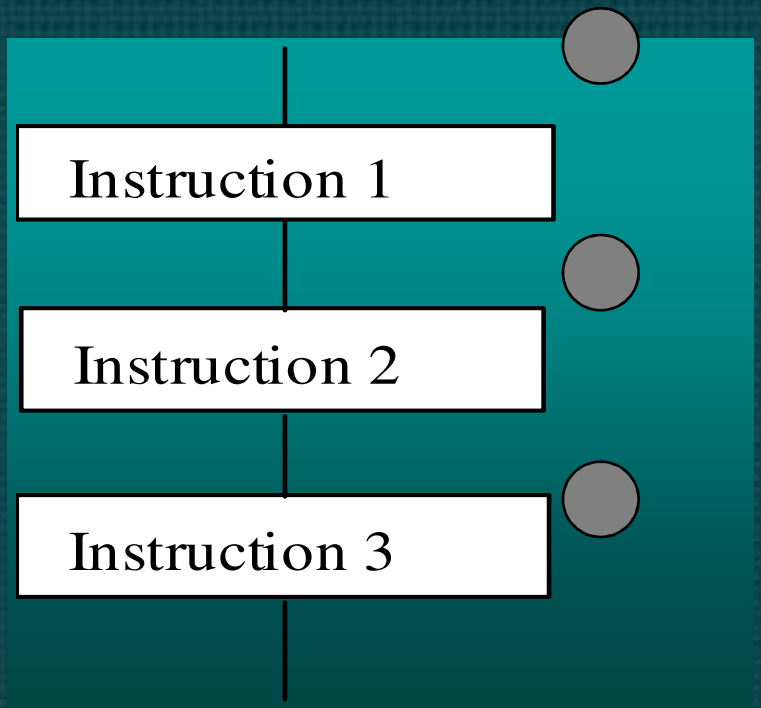
Tests

Gestion du déroulement de l'exécution :

Principes Fondamentaux de la Programmation et Applications en VBA

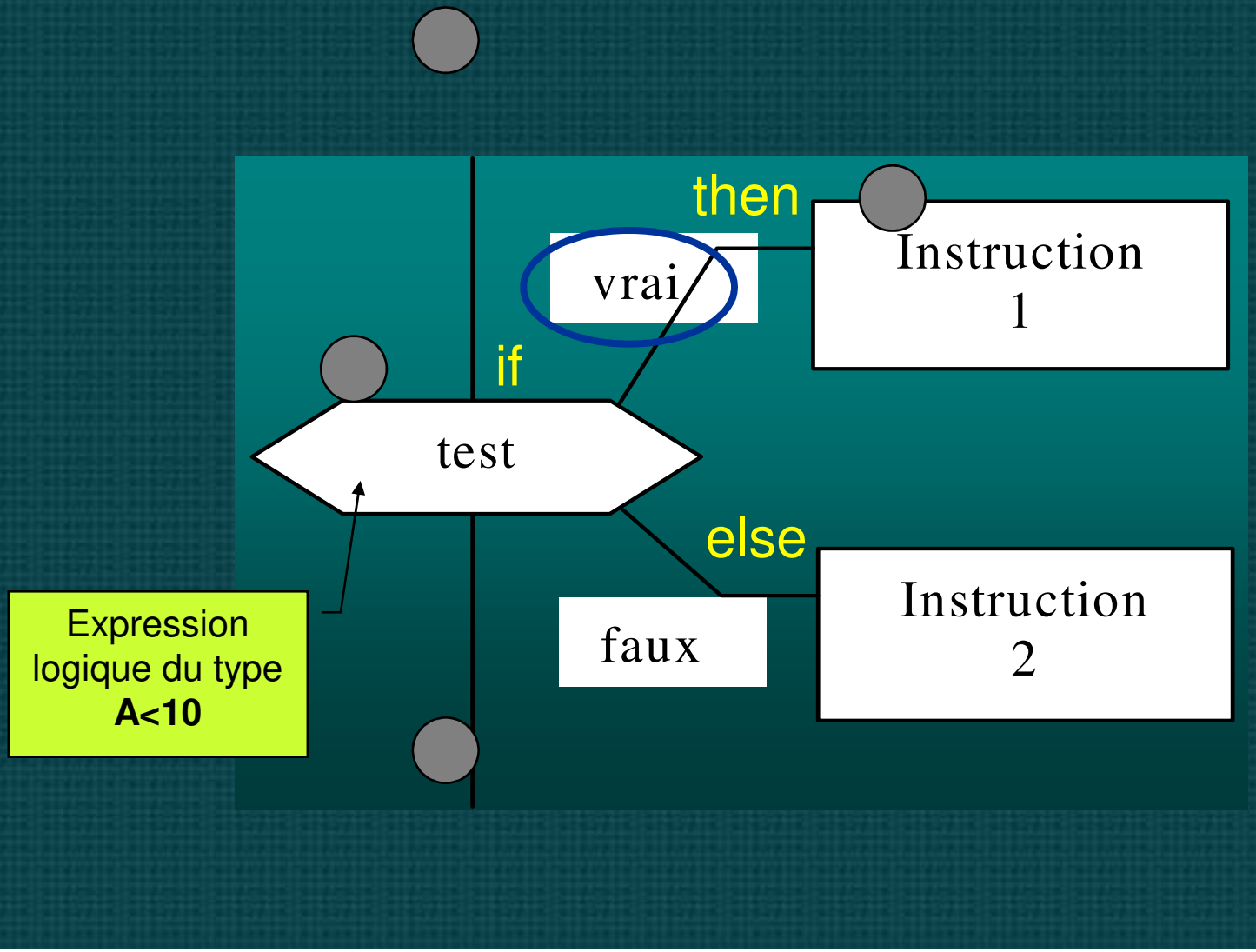
- Par défaut, exécution en **séquence**
- Sinon, possibilité de :
  - Diriger le traitement en fonction d 'un **test** (structure **alternative**)
  - **Répéter** un ensemble d 'instructions (structure **répétitive**)

Séquence:

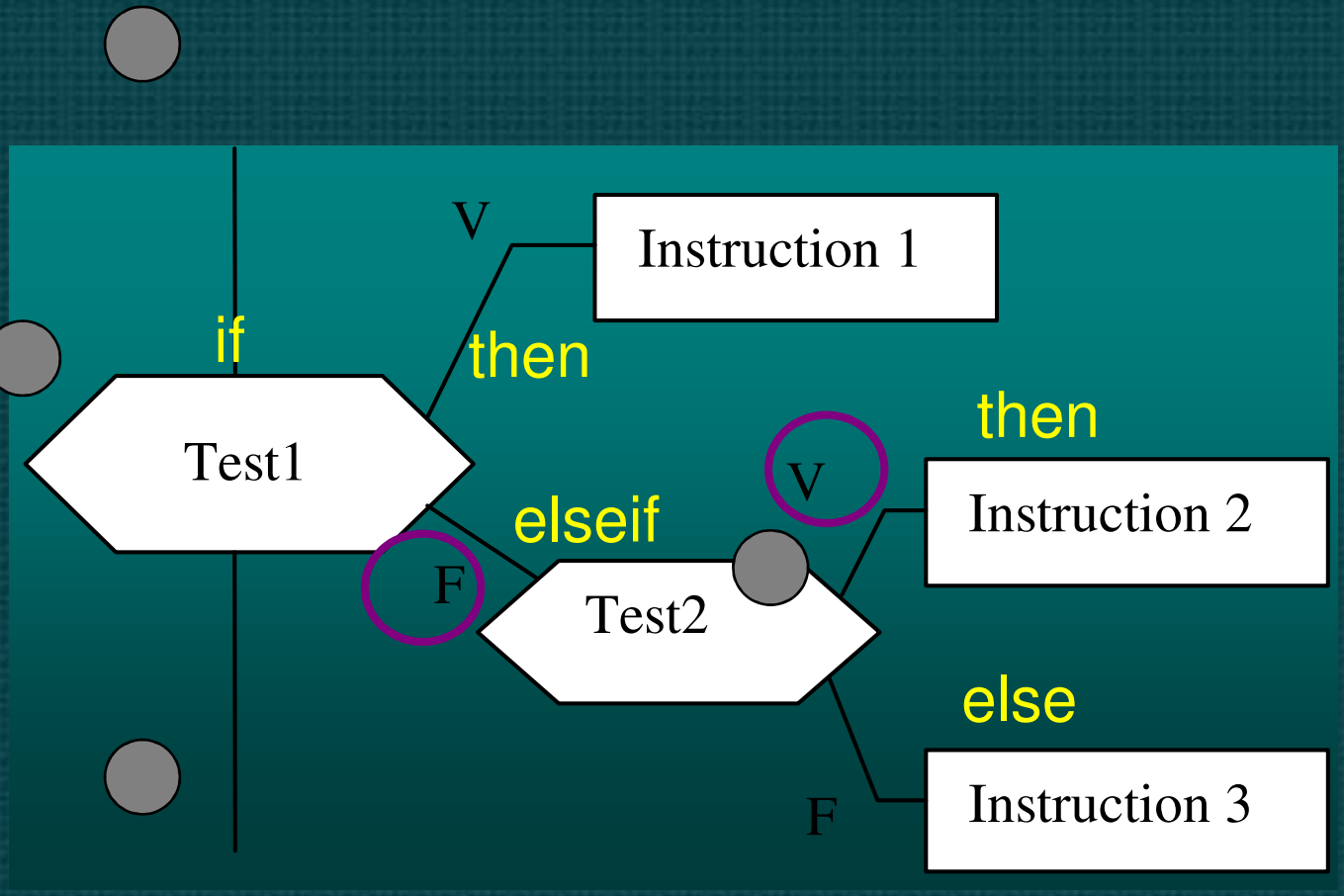


- Les instructions sont exécutées l'une après l'autre
- « Instruction i » peut être une instruction ou un bloc d 'instructions.

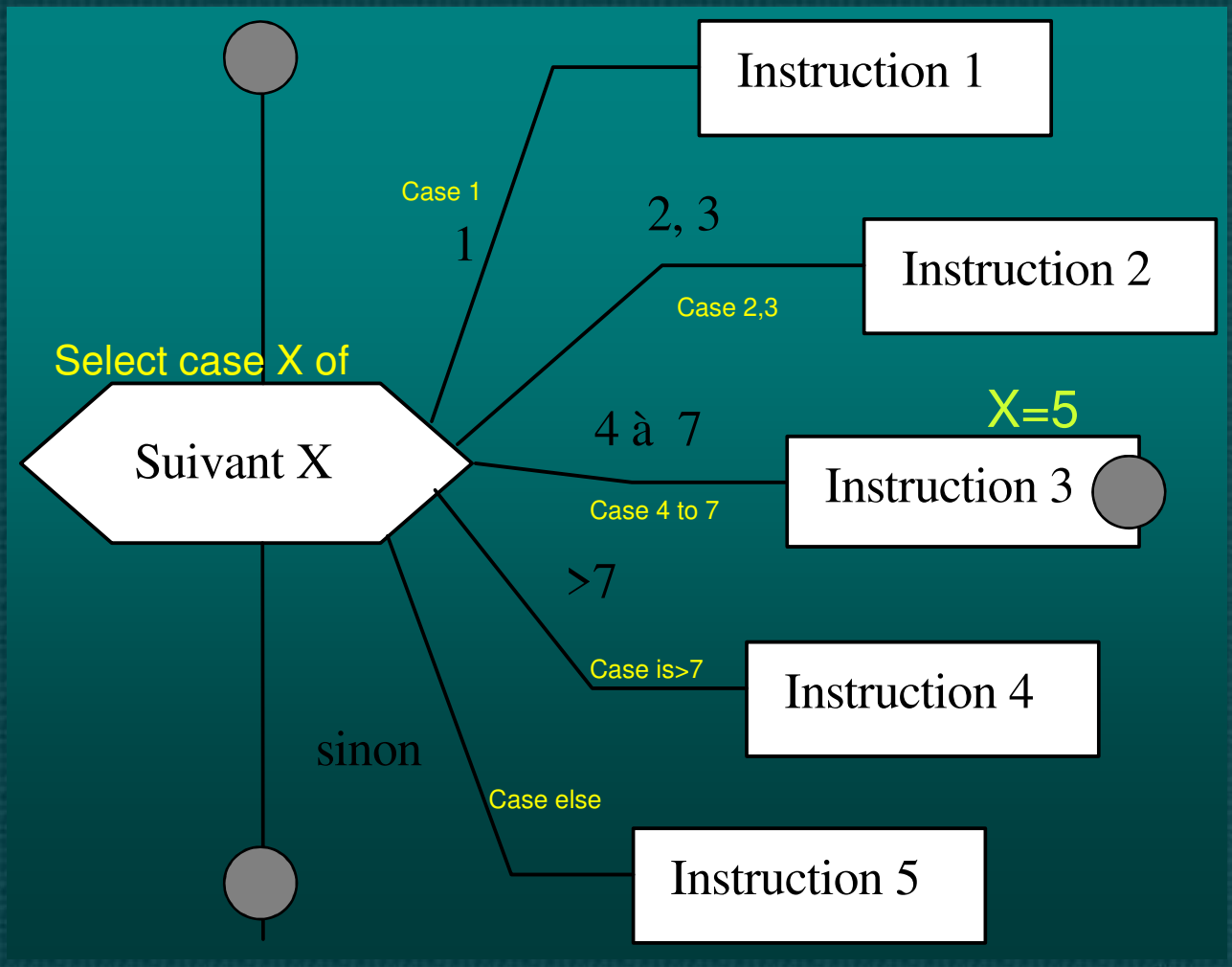
Alternative:



Alternative complexe



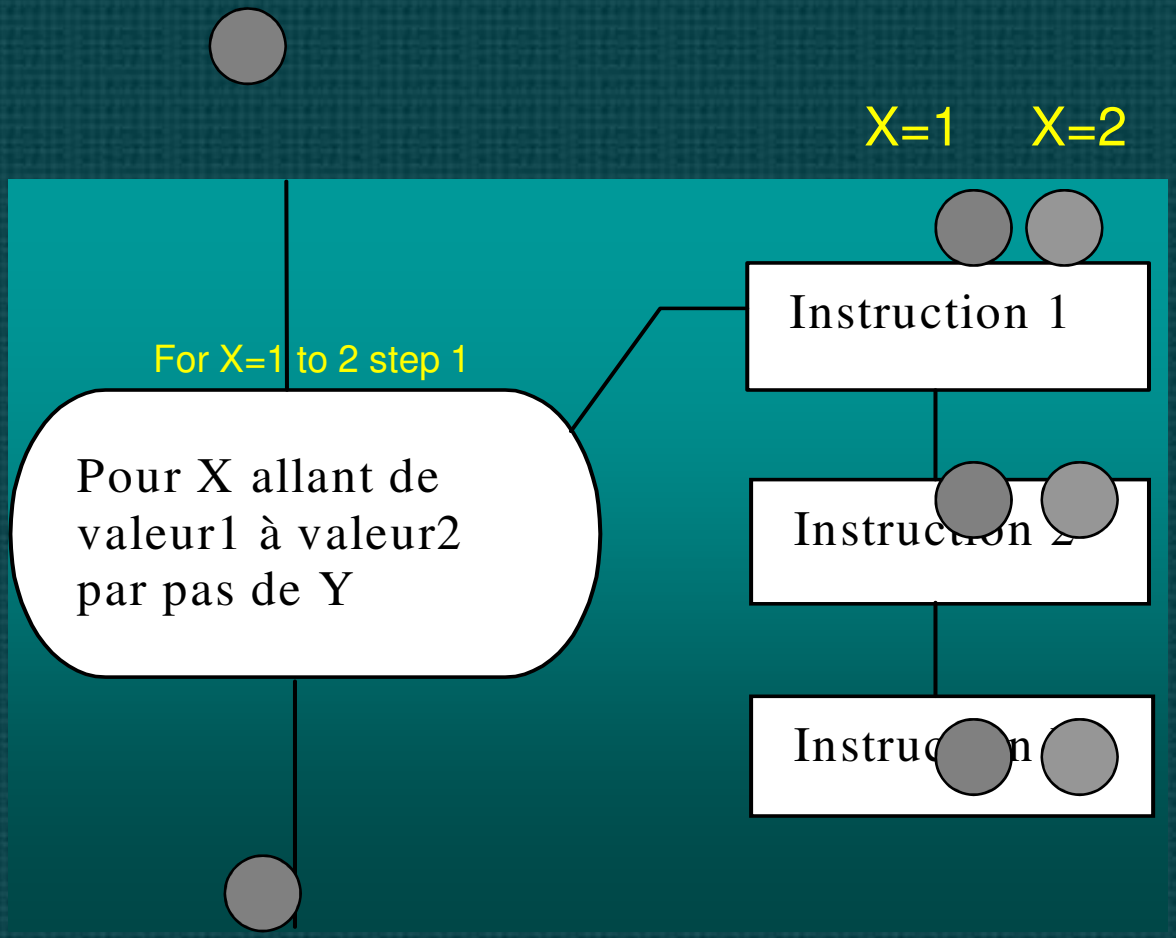
Choix multiple



Structures répétitives

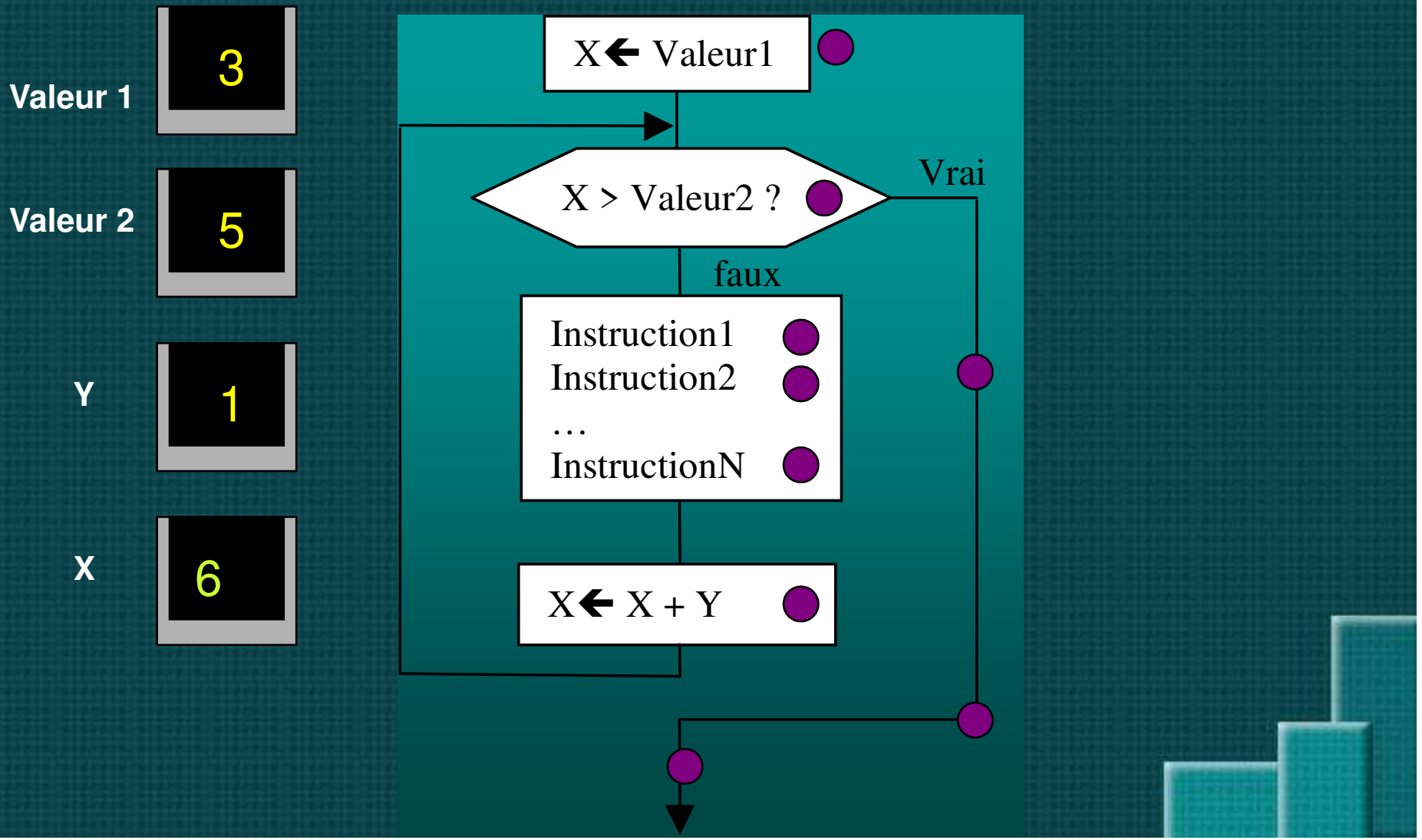
- Appelées aussi « boucles »
- Deux familles :
  - Fixes : nombre de répétitions connu à l'avance
  - Variables : La fin des répétitions est conditionnée par un test effectué à chaque fois.
    Problème de boucles infinies si défaut d'analyse….

Répétition fixe (Pour)



Fonctionnement de « Pour »

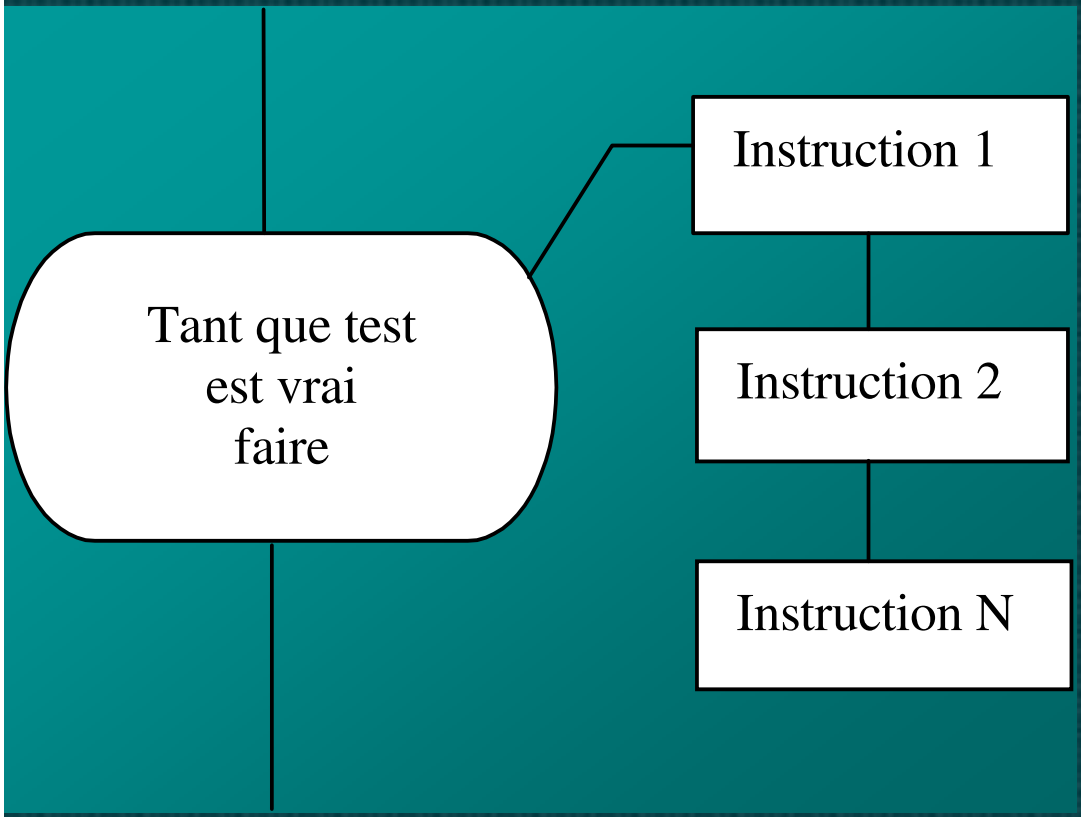
Ex: Pour X variant de 3 à 5 par pas de 1

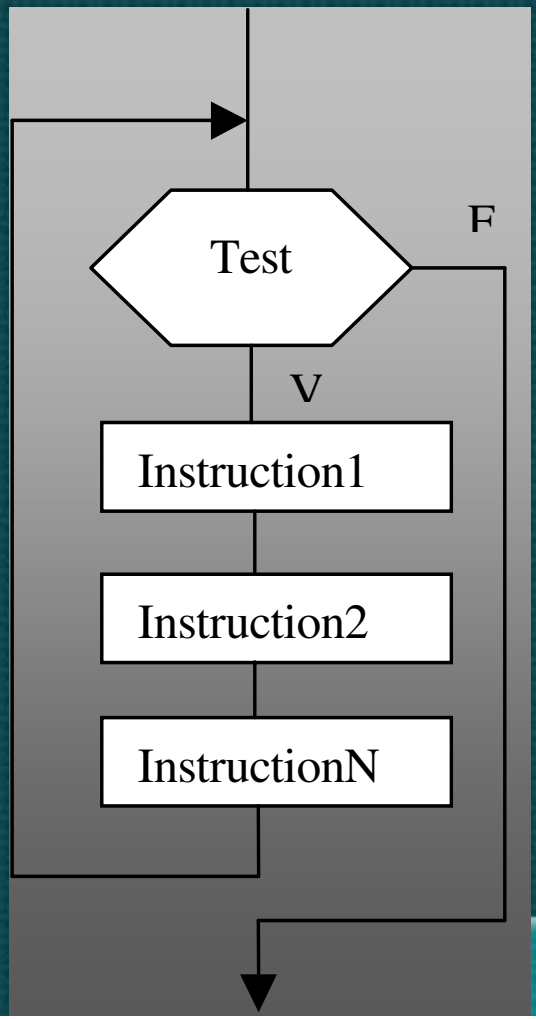


Répétition variable (Tant que)

Test au début

Do while test ….Loop

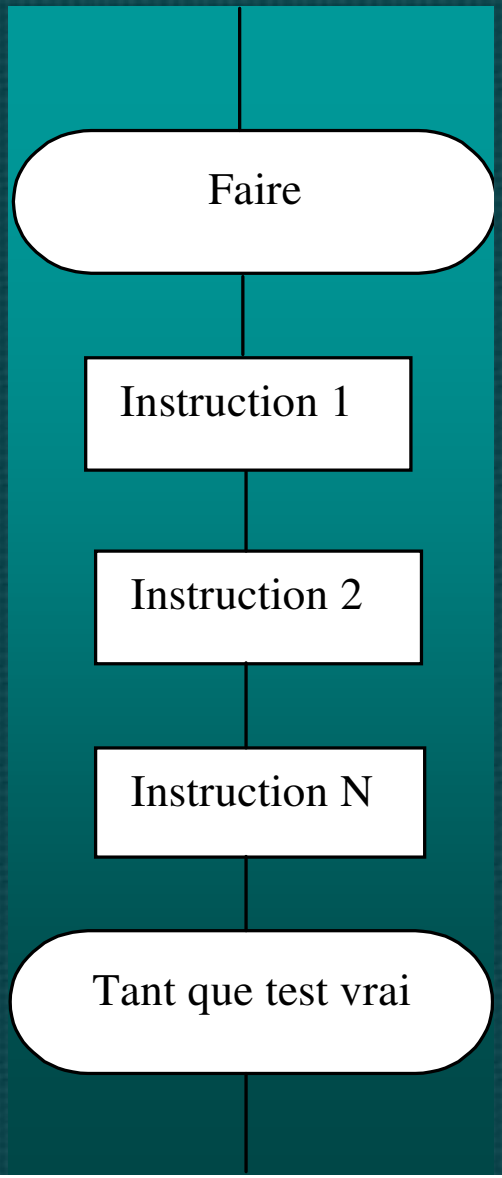


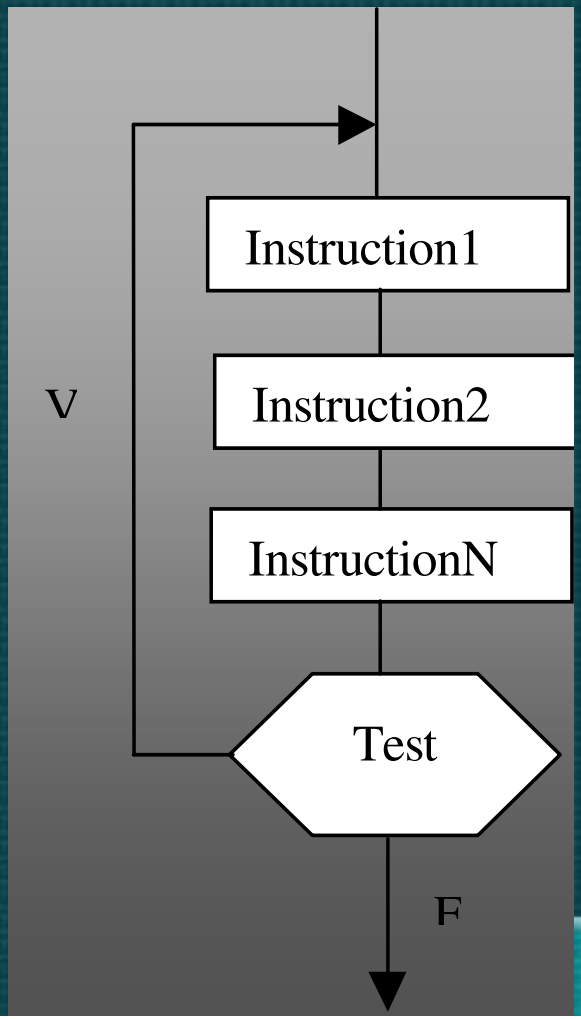


Répétition variable (Tant que)

Test à la fin : la boucle est effectuée au moins une fois

Do…Loop while test

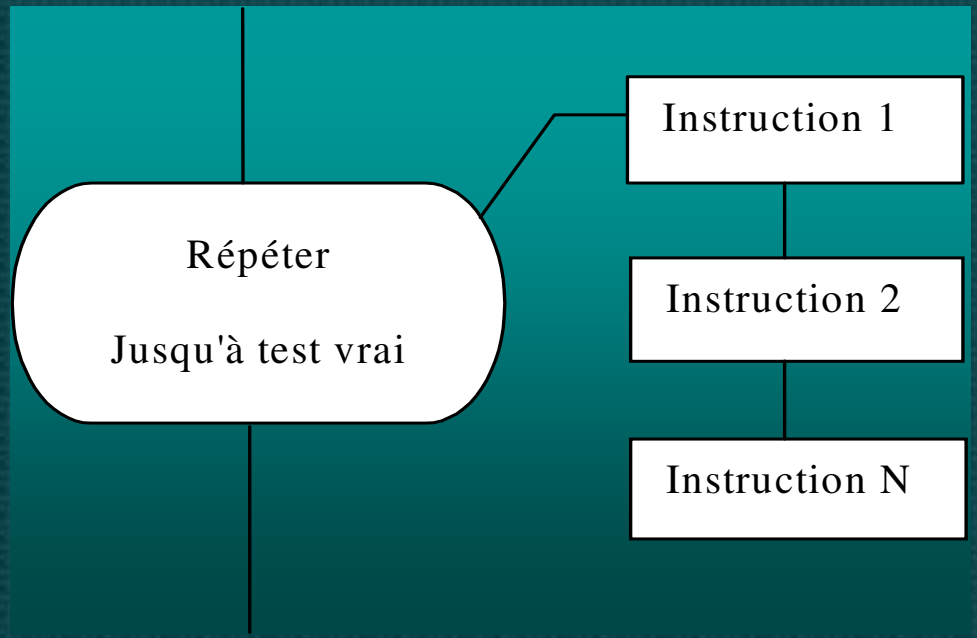


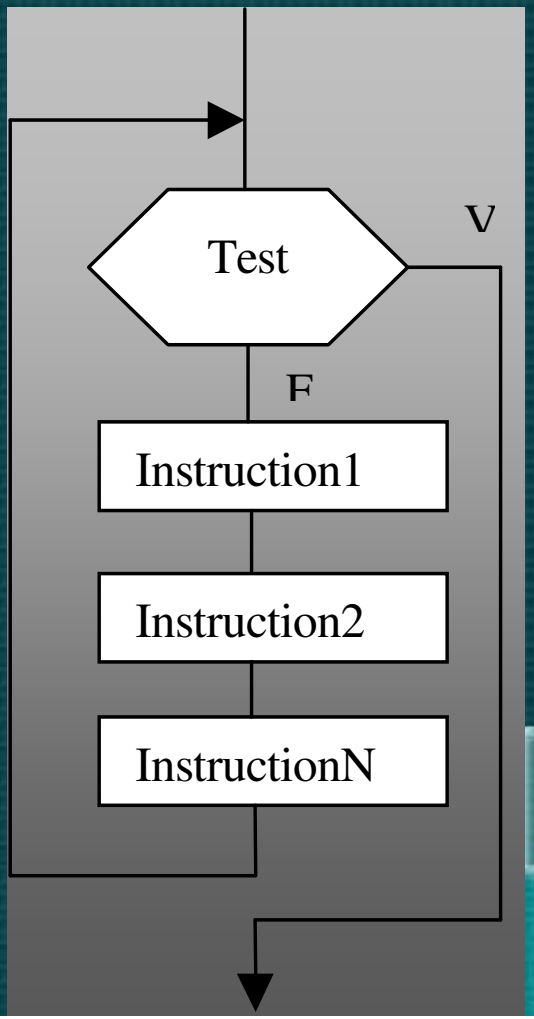


Répétition variable (Jusqu'à)

Test au début

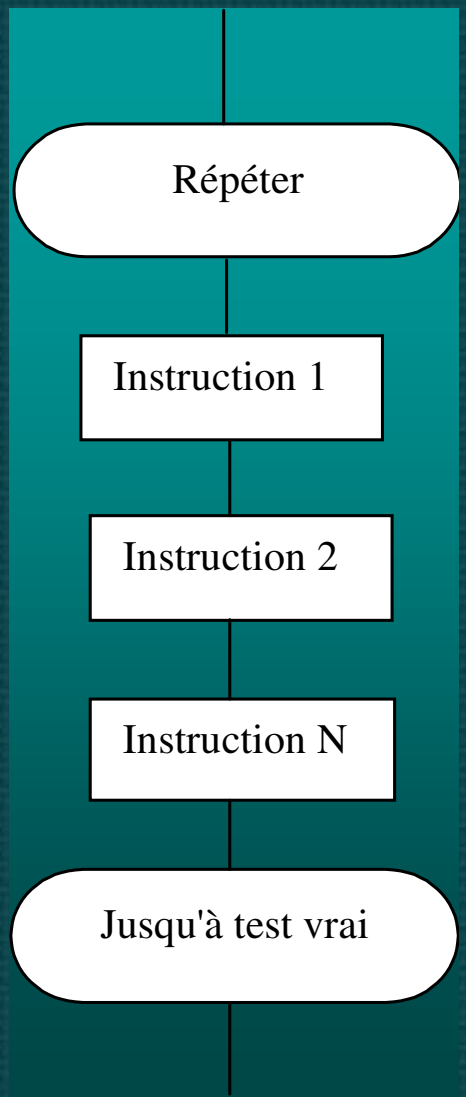
Do Until test...Loop



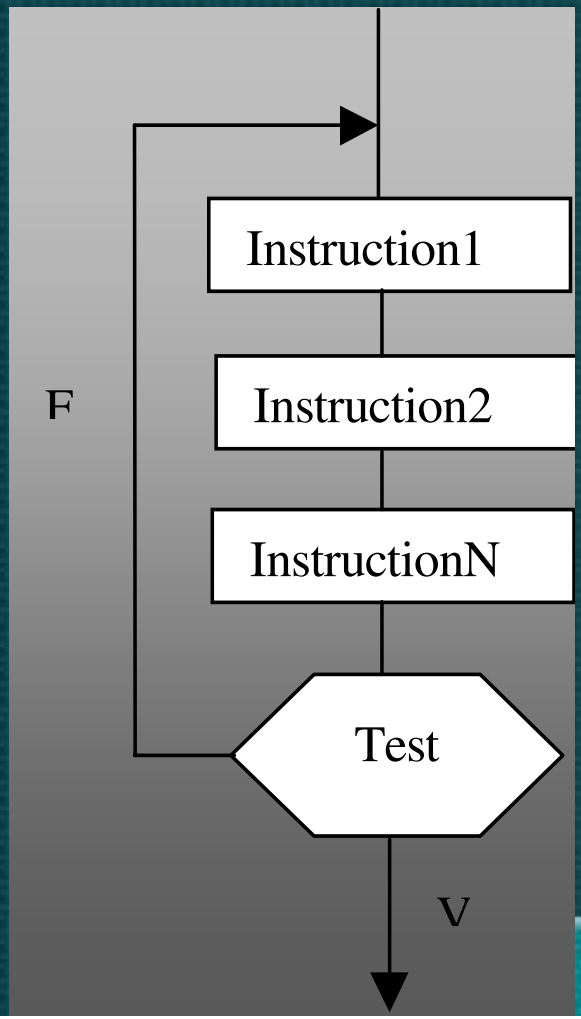


Répétition variable (Jusqu'à)

Test à la fin : la boucle est effectuée au moins une fois



Do…Loop until test

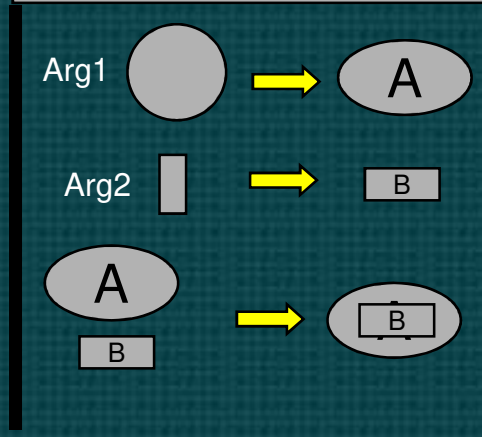


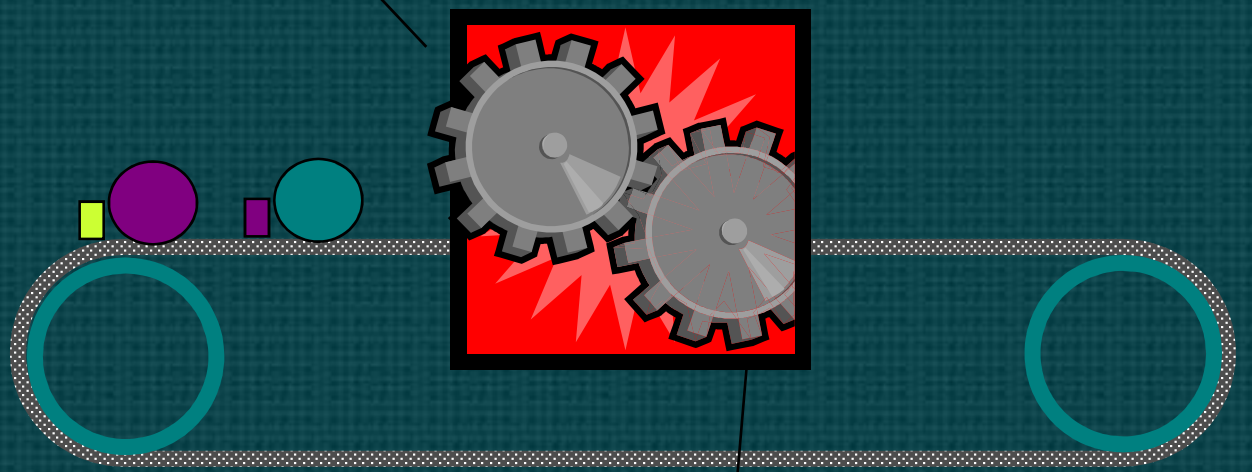
Cas des procédures

- De type SUB si elle produit une ou plusieurs actions
- De type FUNCTION si elle produit un résultat unique que l ’on peut utiliser dans une expression
- Peut nécessiter des ARGUMENTS

Procédures (2) – Fonctions

Description du traitement : algorithme





Arguments :

- Servent à décrire le traitement

- L'ordre de transmission a de l'importance

Outils:
- Variables privées
- Autres procédures

Procédures (2) – Fonctions

Exemple: fonction de conversion en gon d 'un angle en radians

Description du traitement : algorithme

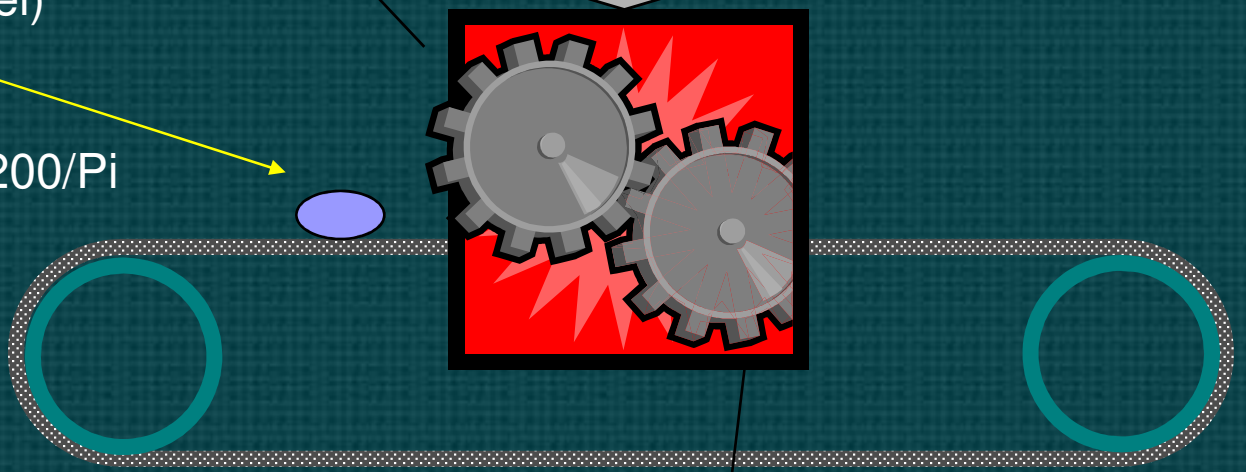
Argument : angle_rad (réel)

Pi ← 4*Atn(1)

Conv_gon ← angle_rad*200/Pi

Résultat réel

Nom: Conv_gon



Outils:
-Variables privées
-Autres procédures

Variable Pi

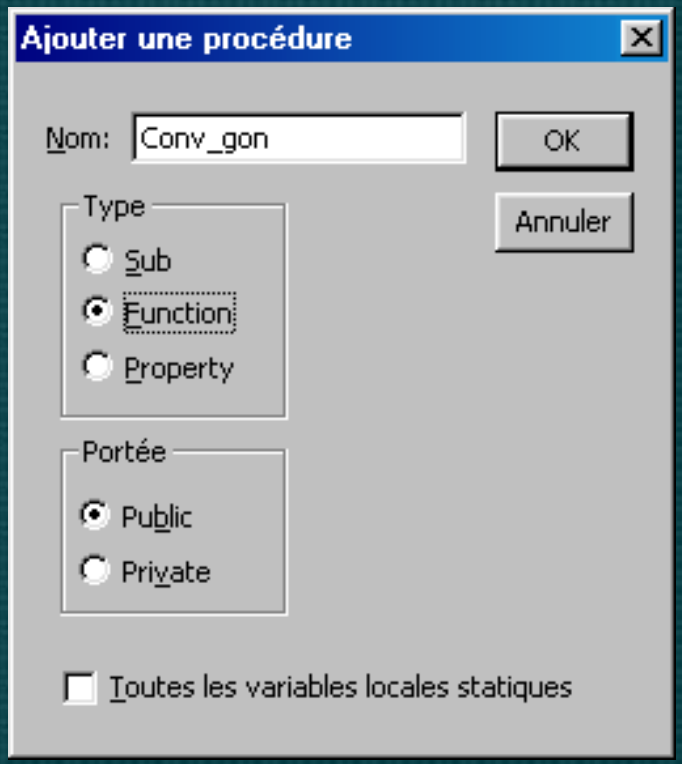
Création d'une fonction (exemple)

Mise en place dans l'environnement VBA

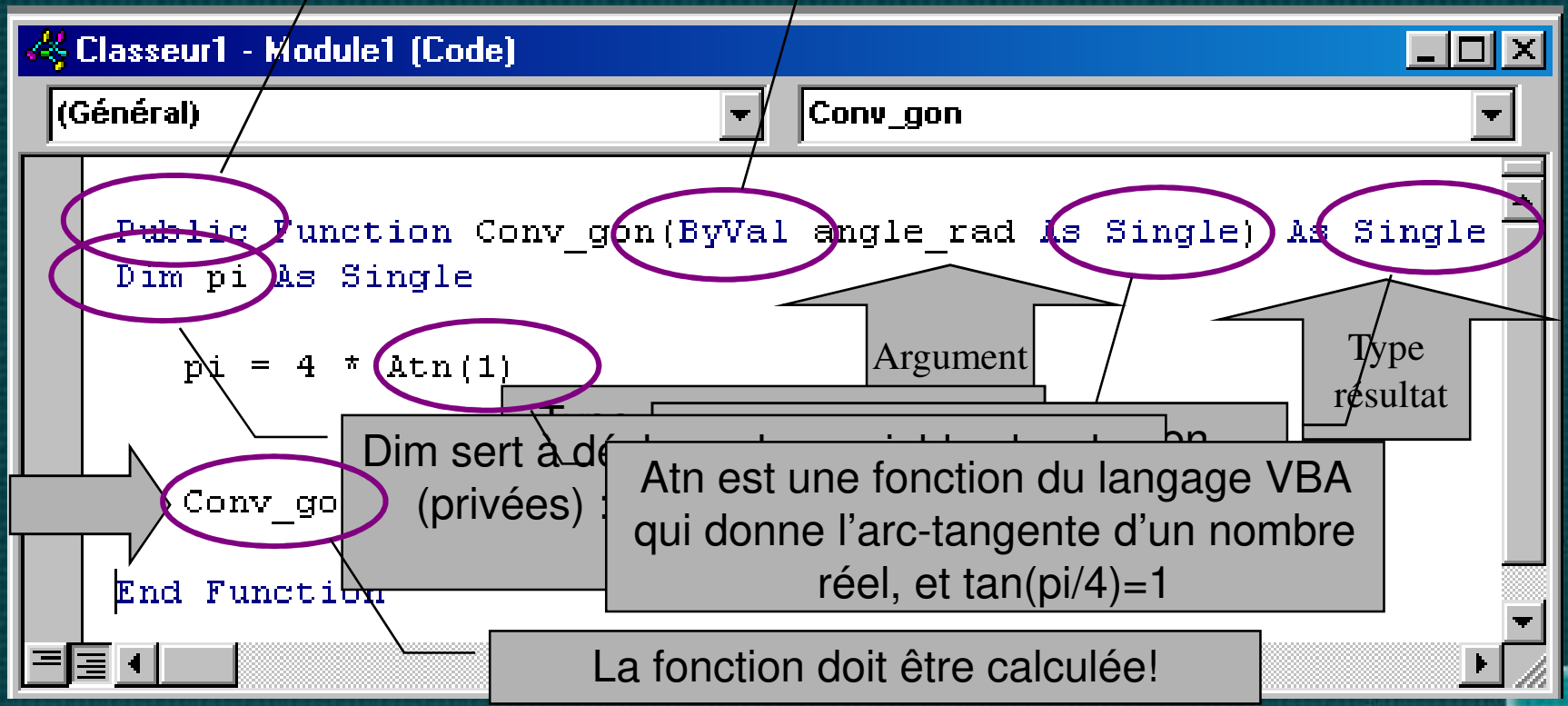
- On lance VBA (ALT-F11)

- On crée un module VBA

- On insère une procédure de type fonction



Création d'une fonction (exemple)



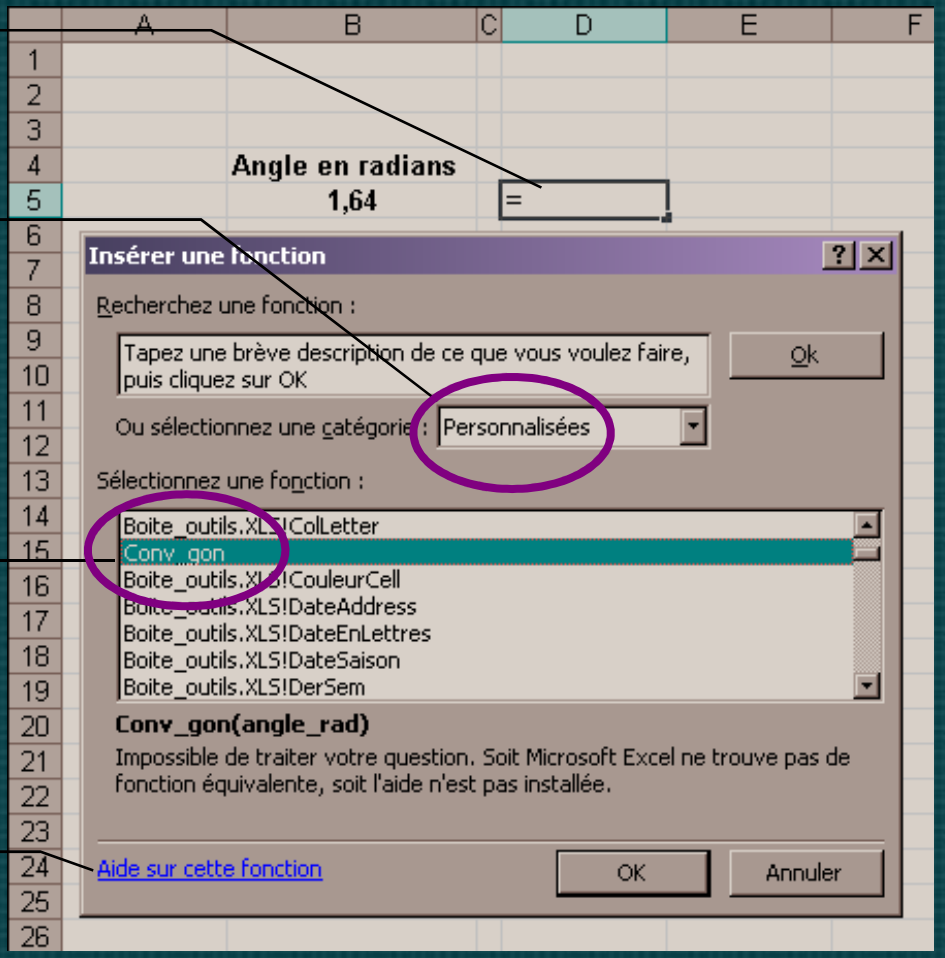
Création d'une fonction (exemple)
Utilisation dans une feuille de calcul

On veut utiliser la fonction ici

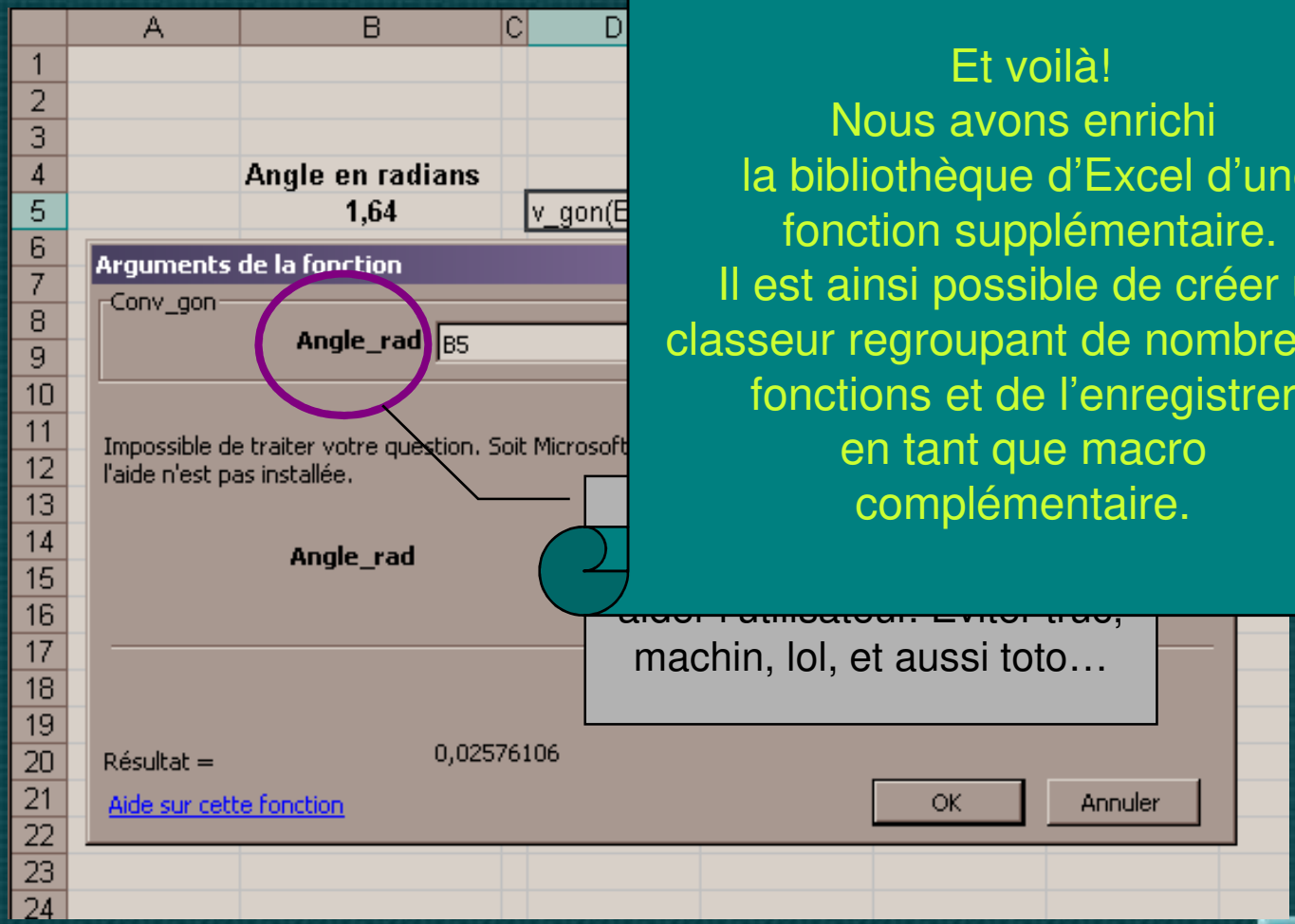
Les fonctions présentes dans des modules VBA sont dans la catégorie « personnalisées »

Remarque: Ne cherchez pas « Boite_outils ». Il s'agit d'un classeur perso truffé de fonctions VBA

Il est possible d'associer un fichier d'aide à la fonction, mais c'est un peu lourd à gérer…



Création d'une fonction (exemple)
Utilisation dans une feuille de calcul



Et voilà!
Nous avons enrichi
la bibliothèque d'Excel d'une
fonction supplémentaire.
Il est ainsi possible de créer un
classeur regroupant de nombreuses
fonctions et de l'enregistrer
en tant que macro
complémentaire.

Création d'une fonction (exemple)

Passer une Variable à une « Function » ou une « Sub » se fait au travers d'un Argument (ou d'Arguments)

1 : Procédure Sub() « appelante » :

```
Public Sub Main()
Dim MaValeur As Integer
   MaValeur = 10
   Call Traiter1(MaValeur) 'MaValeur vaut 100 par la sous procédure Traiter1 où le passage d'arguments se fait ByRef
   MsgBox MaValeur 'MaValeur a été modifiée dans la procédure appelante "Main" et vaut 100
   Call Traiter2(MaValeur) 'MaValeur vaut 1000 par la sous procédure Traiter2 où le passage d'arguments se fait ByVal
   MsgBox MaValeur 'MaValeur n'a pas été modifiée dans la procédure Main et vaut toujours 100
   Call Traiter1(MaValeur) 'MaValeur vaut 1000 par la sous procédure Traiter1 où le passage d'arguments se fait ByRef
   MsgBox MaValeur 'MaValeur a été modifiée dans la procédure appelante "Main" et vaut 1000
End Sub
```

Création d'une fonction (exemple)

2 : Fonctions ou Procédures « appelées »

Deux possibilités de passage d'arguments de la fonction ou procédure à la procédure appelante :

1 : Passage de l'Argument par Référence "ByRef" :

- Il se crée un lien entre l'Argument ("VotreValeur") et la Variable ("MaValeur")
- Toute modification de l'Argument ("VotreValeur") dans la fonction ou la procédure se répercute sur la Variable ("MaValeur") dans la procédure appelante qui est modifiée

```
Public Sub Traiter1(ByRef VotreValeur As Integer)  'Passage d'argument ByRef : Si rien n'était précisé, la passage se ferait aussi "ByRef" par défaut
    VotreValeur = VotreValeur * 10
End Sub
```

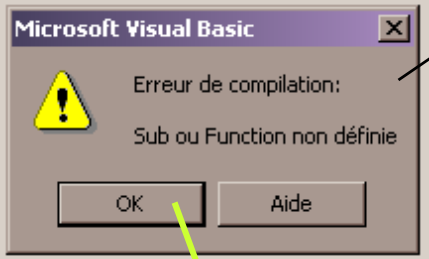
2 : Passage de l'argument par Valeur "ByVal" :

- Aucun lien n'est créé entre l'Argument ("VotreValeur") et la Variable ("MaValeur")
- La fonction ou la Procédure traite l'Argument en interne ("VotreValeur") et affecte les modifications à cet argument
- Puis : Retour à la procédure appelante sans qu'une modification qcq ne soit affectée à la Variable "MaValeur"

```
Public Sub Traiter2(ByVal VotreValeur As Integer)  'Le passage d'argument "ByVal" est précisé ici
    VotreValeur = VotreValeur * 10
End Sub
```

Et si ça se passe mal ?

```
Public Function Conv_gon(ByVal angle_rad As Single) As Single
Dim Pi As Single
  Pi = 4 * Atan(1)
  Conv_gon = angle_rad / 200 * Pi
End Function
```

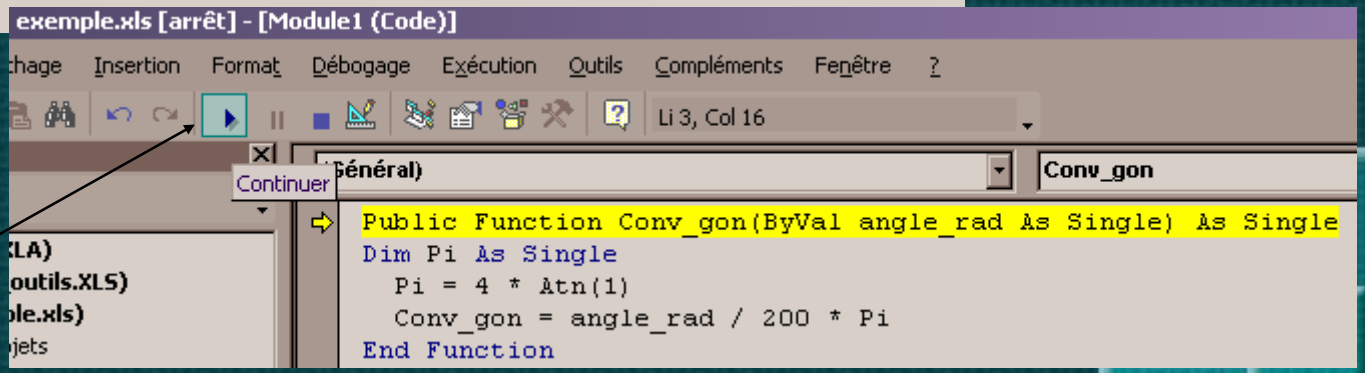


Erreur obtenue lors de la saisie de la fonction

Mode arrêt (Débogage) visible avec la ligne jaune qui indique où en est l'exécution

```
Public Function Conv_gon(ByVal angle_rad As Single) As Single
Dim Pi As Single
  Pi = 4 * Atan(1)
  Conv_gon = angle_rad / 200 * Pi
End Function
```

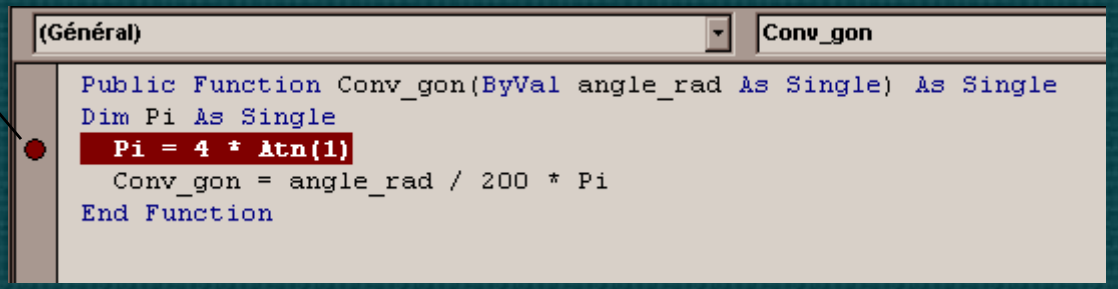
Erreur!

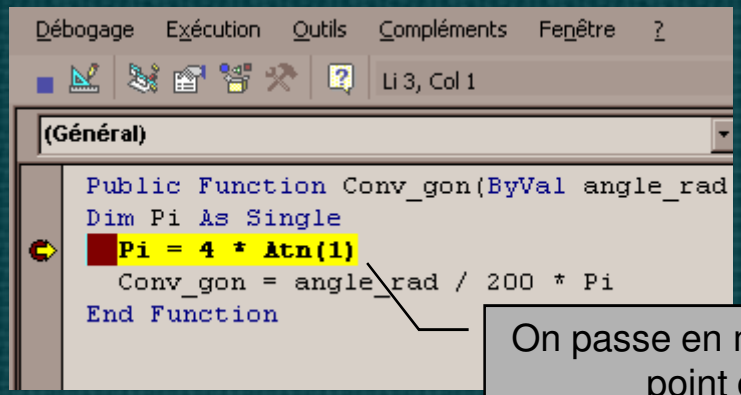


Cliquer sur **Play** pour reprendre l'exécution

Mode débogage (suivi exécution)

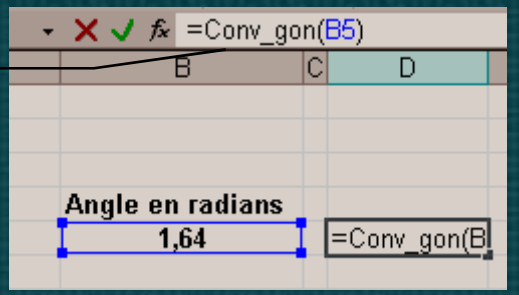
On peut placer un point d'arrêt au niveau d'une instruction



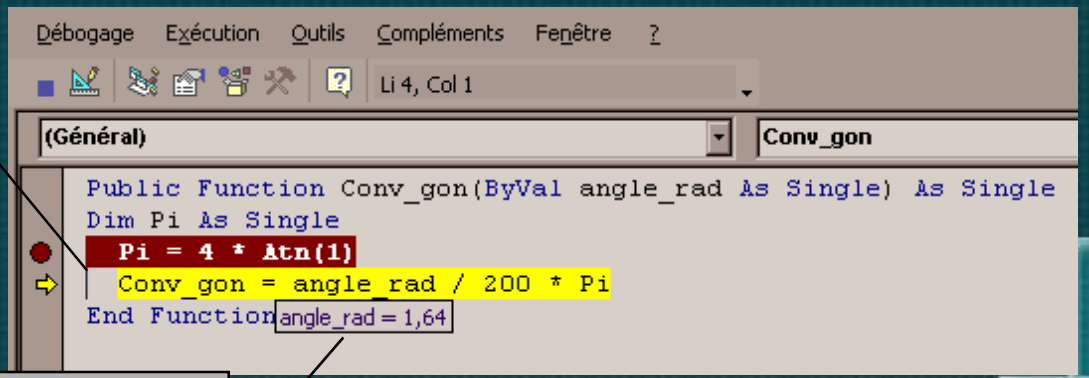


On passe en mode arrêt, au point désiré

Après avoir validé la formule pour forcer un calcul,



Appuyer sur F8 permet de passer à l'instruction suivante



Passer le curseur au dessus d'une variable ou d'un argument permet de vérifier sa valeur.

Pour en apprendre plus..

- Enregistrer des macros et détailler le code.
- Consulter l'aide en ligne
- Utiliser l'explorateur d'objets pour lister les propriétés, les méthodes, etc.

- Et être persévérant…